Lists all files in the repository. The | allows for multiple lines of shell commands (though only one is used here). github.workspace is the path where your repository is checked out.

```
yaml - run: echo "♠ This job's status is ${{ job.status }}."
```

Prints the final status of the job (success, failure, etc.).

This is essentially a demo workflow that shows various features of GitHub Actions:

- How to trigger workflows
- How to use environment variables and contexts (\${{ }}} syntax)
- · How to run commands
- · How to use actions (like checkout)
- How to work with the runner's filesystem

It doesn't do any actual testing or deployment, but it's a good template to learn from when building more complex workflows.

(i) Claude does not have the ability to run the code it generates yet.

HP so it gets variables from github?

Yes, GitHub Actions provides a lot of built-in variables (called "contexts") that you can access using the $\{\{\}\}$ syntax. Here are the main ones used in that workflow:

- github.actor: The username of the person who triggered the workflow
- | github.event_name : What triggered the workflow (like "push", "pull_request", etc.)
- github.ref: The branch or tag name that triggered the workflow
- github.repository: The repository name (like "username/repo-name")
- | github.workspace : The path where your repository is checked out on the runner
- runner.os: The operating system of the runner machine
- job.status: The current status of the job

GitHub provides many more contexts that can be really useful:

- env : Environment variables
- secrets : Repository secrets (for sensitive data)
- strategy : Build matrix information
- matrix : Build matrix values
- steps : Information about the steps that have already run

You can use these variables anywhere in your workflow file to make it dynamic and responsive to different conditions. It's particularly useful for things like:

- Setting different behavior for different branches
- Using repository secrets for deployments
- Creating conditional steps based on event types
- Accessing commit information

Would you like to see some practical examples of how these variables can be used in more complex workflows?



There are several places to store variables in GitHub Actions, depending on their sensitivity and scope:

1. Repository Secrets (for sensitive data)

```
yaml

# Access in workflow
${{ secrets.MY_SECRET }}
```

- Stored in: Repository Settings \rightarrow Secrets and Variables \rightarrow Actions
- Best for: API keys, passwords, tokens
- Encrypted and only masked in logs
- 2. Repository Variables (for non-sensitive data)

```
yaml

# Access in workflow
${{ vars.MY_VARIABLE }}
```

- Stored in: Repository Settings \rightarrow Secrets and Variables \rightarrow Actions
- Good for: Configuration values, environment names, URLs
- 3. Environment Variables (in the workflow file)

```
yaml

name: Workflow
env:

SERVER_NAME: production-server
DATABASE: my-db

jobs:
build:
env:
NODE_ENV: production
```

4. Environment-specific Secrets/Variables

- • Stored in: Repository Settings → Environments → Select Environment → Add Secret/Variable
- Good for: Having different values for staging/production

 $\label{problem:example combining these approaches:} Example combining these approaches:$

```
name: Deploy
env:

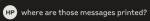
APP_NAME: my-app

jobs:
deploy:
environment: production
env:
NODE_ENV: production
steps:
- name: Deploy
run: |
echo "Deploying $APP_NAME"
echo "Using database ${{ secrets.DATABASE_URL }}"
echo "Environment type: ${{ vars.ENVIRONMENT_TYPE }}"
```

The general rule is:

- Secrets for sensitive data (never visible in logs)
- Variables for non-sensitive configuration
- Environment variables in workflow for build-specific settings
- Environment-specific values when you need different values per environment

 $\ensuremath{\ensuremath{\mbox{\scriptsize 0}}}$ Claude does not have the ability to run the code it generates yet



These messages are printed in the GitHub Actions workflow logs, which you can see in a few places:

1. GitHub Actions Tab

